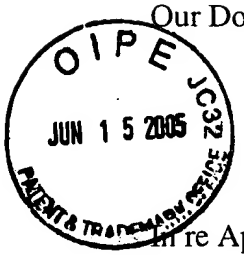
Our Docket No.: 42P7512



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Orna Etzion

Application No.: 09/676,175

Filed: September 29, 2000

For: A Method and Apparatus for Generating an Expected Top of Stack During Instruction Translation

Examiner: Meonske, Tonia L.

Art Group: 2183

Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION UNDER 37 CFR 1.131 IN SUPPORT OF PRIOR INVENTION

Sir :

I, Orna Etzion, declare:

1. I am an inventor of the claims of the above-captioned patent application ("the Application") and an inventor of the subject matter described therein.

2. Prior to June 16, 2000, the filing date of U.S. Patent No. 6,725,361 cited in an Office Action mailed June 01, 2004, the invention claimed in the Application had been conceived and reduced to practice in the United States.

3. Attached Exhibit A is a redacted copy of an invention disclosure form describing the design of the A Method and Apparatus for Generating an Expected Top of Stack During Instruction Translation, and establishes that the subject matter claimed in the Application had been reduced to practice in the United States prior to June 16, 2001. Exhibit A (the invention disclosure) describes the operation of generating an expected top of stack during instruction translation, as is described and claimed in our application. More specifically, the figure on page 3 of Exhibit A illustrates the features of claim 1. The features of "translating a first block of instructions executable in a first processor architecture, into a translated first block of instructions executable in a second processor architecture, said translated first block of instructions operationg with a stack of data entry positions" is shown in the top five blocks of the figure with the transformation of 'Code Block L1' to 'Code Block L2'. The feature of "generating an expected Top of Stack (TOS) position in said stack for said first block of code" is shown in the top five blocks of the figure by the entry condition 'Expected TOS'. The feature of "adding at least one instruction to said translated first block of instructions to determine if said first expected TOS is equal to an actual TOS at a time of executing said translated first block of instructions" is shown in the top five blocks of the figure in the translated pseudo-code sections of 'Code Block L1' and 'Code Block L2'.

4. The subejct matter claimed in the application was actually reduced to practice prior to June 16, 2000 because the technique claimed in Exhibit A had been successfully implemeted before this date, as noted on page 1, paragraph 3 in Exhibit A.

that these statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application of any patent issuing thereon.

Dated: June 6th, 2005

Orna Etzion

ML Comm / MPG / MPL file

INTEL CONFIDENTIAL

INTEL INVENTION DISCLOSURE

JUL - 7 1999

LEGAL ID# 12073 Exhibit A DATE: July 7, 1999

It is important to provide accurate and detailed information on this form. The information will be used to evaluate your invention for possible filing as a patent application. When completed, please return this form to the Legal Department at RN4-01. If you have any questions regarding this form or to whom it should be forwarded, please call 765-1369, 696-2851 or 554-3996.

1. Inventor(s):

Name: Oma Etzion SS# N/A

Empl. No. 10122359 Dept.# 6985 Phone 4865-5720 M/S: IDC-1 D

Home Address: 5 Kariv st. Haifa, Israel

Citizenship: Israel Supervisor* Yaron Sheffer M/S: IDC-1 D

Group Name: MPL Division Name: MPG

Name: ? SS# N/A

Empl. No. Dept.# 6985 Phone ? M/S: IDC-1 D

Home Address:

Citizenship: Israel Supervisor* Yaron Sheffer Phone 4865-5759 M/S: IDC-1 D

Group Name: ML Division Name: MPG

2. Title of Invention: A method for efficiently maintaining synchronization of a simulated circular-stack of registers during binary translation.

3. Stage of development, i.e. % complete, and relation of technology to the following product/process:
The technique has been implemented in a dynamic IA32→IA64 binary translator, which is currently a research project, for floating-point stack simulation.

4. (a) Has a description of your invention been, or will it shortly be, published outside Intel:

NO: _____ YES: X DATE WAS OR WILL BE PUBLISHED: 10/99

If YES, was the manuscript submitted for pre-publication approval? YES: X NO: _____

(b) Has your invention been used/sold or planned to be used/sold by Intel or others?

NO: _____ YES: X DATE WAS OR WILL BE SOLD: may be used in future implementations of IA64, not yet on plan of record.

5. If invention conceived, or constructed during performance of a government or third party contract, please check here ____and give the contract name and number _____.

6. Please attach a page to this form, DATED AND SIGNED BY ONE INVENTOR (PREPARER), to provide an abstract of your invention, and include the following information in your abstract:

(a) State general purpose(s) of your invention;
(b) Describe advantage(s) of your invention over what is done now;
(c) Describe essential element(s) or key to your invention; and
(d) Value of your invention to Intel (how will it be used?).

*HAVE YOUR SUPERVISOR READ, DATE AND SIGN COMPLETED FORM

DATE: ? SUPERVISOR: Yaron Sheffer

BY THIS SIGNING, I (SUPERVISOR) ACKNOWLEDGE THAT I HAVE READ AND UNDERSTAND THIS DISCLOSURE, AND RECOMMEND THAT THE HONORARIUM BE PAID.

General purpose of the invention

The purpose of this invention is to efficiently maintain synchronization of a simulated circular register stack. The invention may be valuable for binary translation, from source computer architecture that contains such a stack, to a target architecture that supports a flat register file. The invention may be used in dynamic or static binary-translators, as well as in architectural simulators or virtual-machine implementations using similar, code-generation-based, techniques. In particular, the invention provides a significant performance advantage when translating Intel Architecture floating-point code to any other architecture.

Advantages of the invention over what is done now

The invention is significantly faster than any known alternative.
Emulating a stack rotation by multiple move operations need to perform those moves for any stack push or pop. The number of the required moves per occurrence is the size of the stack, and they contain a lot of internal dependencies. the proposed invention, the rotation moves are performed only on extremely rare cases.
Emulating a stack in memory suffers from a great load-store overhead, which the proposed invention avoids.

Essential elements or key to the invention

The following section demonstrates the key elements of the invention using, as an example, an IA32→IA64 binary translator. The relevant aspect is the emulation of IA32 floating-point (FP) register stack, using the flat FP register-file of an IA64 target machine.

References to the eight physical FP-registers of the Intel IA32 architecture are always stack-relative. The mapping between stack-relative references and physical registers changes dynamically. For example, the physical registers corresponding to ST(0) before and after executing an FLD instruction are different, since FLD pushes a value onto the FP-stack.

However, in the vast majority of practical cases, multiple run-time entries to the same code block repeat the same stack-depth at entrance. Speculating the state at the entry point allows an effective *static* mapping between any IA32 FP register-references in the block and the corresponding IA64 FP-registers. To take advantage of such a speculative approach, the following mechanisms are supported:

1. Stack depth speculation – effectively guessing the run-time stack state at all or almost all entries to the block. The speculation is done prior to the block translation. Dynamic translator uses the 1st run-time entry state (which is already known when the block is reached). Static translator has to perform code analysis and walk-through to predict the entrance state effectively.
2. Tracking the speculation realization – keeping the actual run time stack state and verifying that the speculative assumption (taken at the translation of the block) is indeed true at each run-time entry. The actual stack depth is updated at the end of the block execution, which is a single operation that reflects the overall effect of the entire block. If the block is balanced (same number of pushes and pops), this code is eliminated. At the beginning of each block, a checking code is executed, that compares the assumed (speculated) stack depth with the actual one.
3. Recovery mechanism – ensure correct operation when the check fails. The recovery is achieved by actual rotation (copy of register values), so the actual top-of-stack moves to fit the expected one. The block code remains as is. This method of recovery ensures that the penalty does not propagate: When control is transferred to the next block, the correction is already done, and the stack-depth expected by the next block matches the actual depth.

Note: This invention disclosure does not describe how stack exception conditions are detected. The solution to that problem is covered by another patent disclosure.

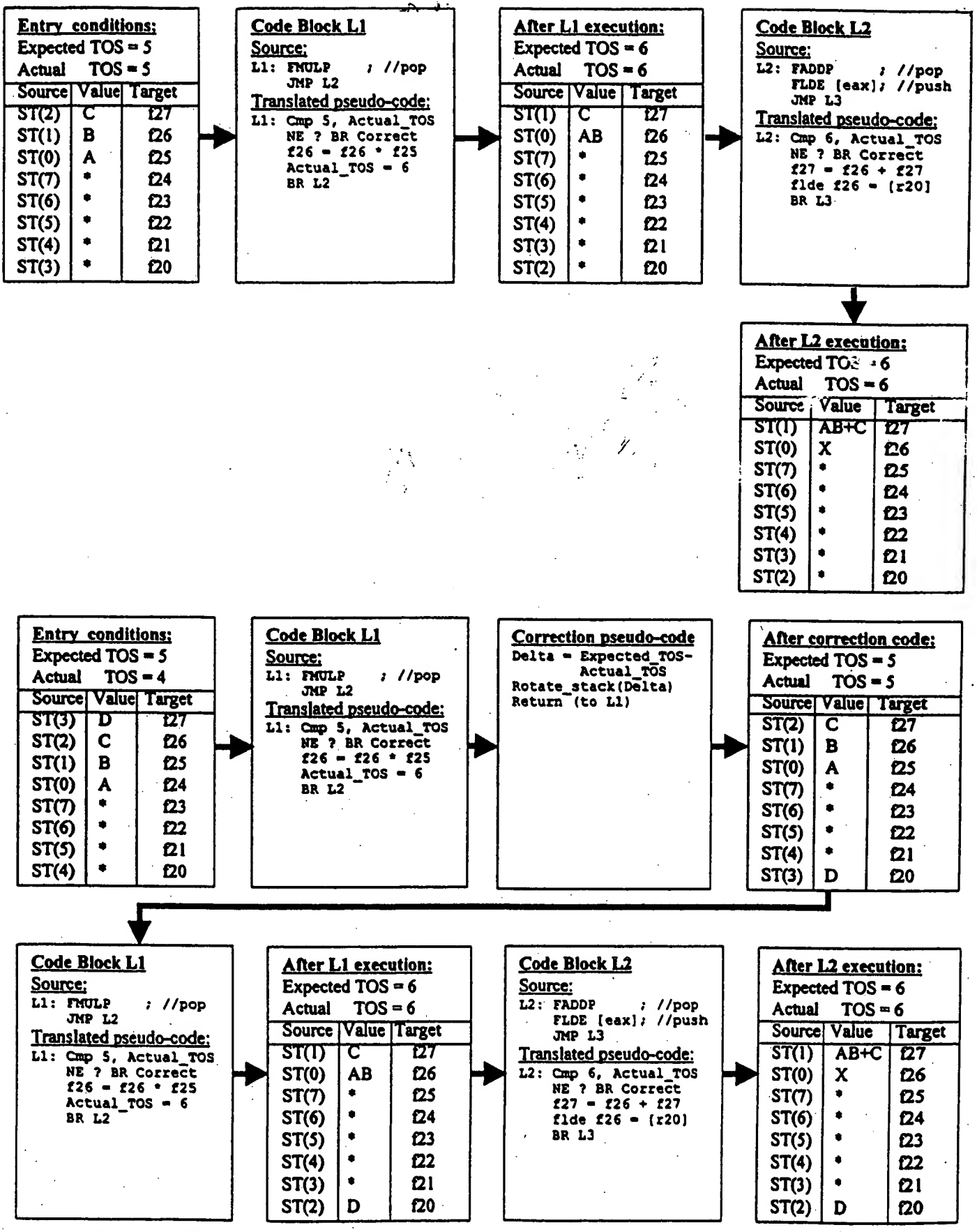
Example

The example in the following page consists of 2 very simple floating-point blocks. It shows the behavior of the translation mechanism at the regular case (when the expected Top-Of-Stack equals the actual one), and on the special case (when they are different). Note that L2 block is balanced, hence no update of the actual TOS value is done at its epilogue. Also note that the correction done for L1 (on the special case) does not affect the normal flow at L2. The Actual TOS value is best held in a global integer register (but not necessarily).
As already stated, although the example refers to IA32→IA64 translation, the invention principles are applicable to any other case of emulating a rotating stack by a static register file.

Value of the invention to Intel: how will it be used?

This invention is valuable to Intel because it can be use to significantly speed up the floating-point performance of IA32→IA64 dynamic binary translation. Such a project currently exists as a research project, but the technology is expected to eventually enter a commercial product of strategic importance to Intel.

Example


Entry conditions:
Expected TOS = 5
Actual TOS = 5
Source | Value | Target
ST(2) | C | f27
ST(1) | B | f26
ST(0) | A | f25
ST(7) | * | f24
ST(6) | * | f23
ST(5) | * | f22
ST(4) | * | f21
ST(3) | * | f20
Code Block L1
Source:
L1: FMULP ; //pop
JMP L2
Translated pseudo-code:
L1: Cmp 5, Actual_TOS
NE ? BR Correct
f26 = f26 * f25
Actual_TOS = 6
BR L2
After L1 execution:
Expected TOS = 6
Actual TOS = 6
Source | Value | Target
ST(1) | C | f27
ST(0) | AB | f26
ST(7) | * | f25
ST(6) | * | f24
ST(5) | * | f23
ST(4) | * | f22
ST(3) | * | f21
ST(2) | * | f20
Code Block L2
Source:
L2: FADDP ; //pop
FLDE [eax]; //push
JMP L3
Translated pseudo-code:
L2: Cmp 6, Actual_TOS
NE ? BR Correct
f27 = f26 + f27
flde f26 = [r20]
BR L3
After L2 execution:
Expected TOS = 6
Actual TOS = 6
Source | Value | Target
ST(1) | AB+C | f27
ST(0) | X | f26
ST(7) | * | f25
ST(6) | * | f24
ST(5) | * | f23
ST(4) | * | f22
ST(3) | * | f21
ST(2) | * | f20
Entry conditions:
Expected TOS = 5
Actual TOS = 4
Source | Value | Target
ST(3) | D | f27
ST(2) | C | f26
ST(1) | B | f25
ST(0) | A | f24
ST(7) | * | f23
ST(6) | * | f22
ST(5) | * | f21
ST(4) | * | f20
Code Block L1
Source:
L1: FMULP ; //pop
JMP L2
Translated pseudo-code:
L1: Cmp 5, Actual_TOS
NE ? BR Correct
f26 = f26 * f25
Actual_TOS = 6
BR L2
Correction pseudo-code
Delta = Expected_TOS-
Actual_TOS
Rotate_stack(Delta)
Return (to L1)
After correction code:
Expected TOS = 5
Actual TOS = 5
Source | Value | Target
ST(2) | C | f27
ST(1) | B | f26
ST(0) | A | f25
ST(7) | * | f24
ST(6) | * | f23
ST(5) | * | f22
ST(4) | * | f21
ST(3) | D | f20
Code Block L1
Source:
L1: FMULP ; //pop
JMP L2
Translated pseudo-code:
L1: Cmp 5, Actual_TOS
NE ? BR Correct
f26 = f26 * f25
Actual_TOS = 6
BR L2
After L1 execution:
Expected TOS = 6
Actual TOS = 6
Source | Value | Target
ST(1) | C | f27
ST(0) | AB | f26
ST(7) | * | f25
ST(6) | * | f24
ST(5) | * | f23
ST(4) | * | f22
ST(3) | * | f21
ST(2) | D | f20
Code Block L2
Source:
L2: FADDP ; //pop
FLDE [eax]; //push
JMP L3
Translated pseudo-code:
L2: Cmp 6, Actual_TOS
NE ? BR Correct
f27 = f26 + f27
flde f26 = [r20]
BR L3
After L2 execution:
Expected TOS = 6
Actual TOS = 6
Source | Value | Target
ST(1) | AB+C | f27
ST(0) | X | f26
ST(7) | * | f25
ST(6) | * | f24
ST(5) | * | f23
ST(4) | * | f22
ST(3) | * | f21
ST(2) | D | f20